

## REMARKS

Reconsideration of this application, based on this amendment and these following remarks, is respectfully requested.

Claims 2 through 17 remain in this case. Claims 2, 9, 11, 16, and 17 are amended.

Amendment to Figure 1 is proposed in this amendment, in light of the objections to the drawings made in the Office Action. Specifically, the reference numerals for the L1, S1, M1, D1 elements are reversed, and the reference numerals for the L2, S2, M2, D2 elements are reversed, to be consistent with the specification at pages 8 and 9. In addition, reference numeral 2 is displayed in a different manner, specifically by an underlined 2 within the box, to distinguish its element from the overall system referred to by reference numeral 1. Applicant submits that this amendment to Figure 1 overcomes the objections raised by the Examiner.

Claim 11 was rejected under §112, ¶2 as indefinite for failing to particularly point out and distinctly claim the subject matter of the invention, specifically for lacking antecedent basis for its recitation of the level two cache. Claim 11 is amended to overcome the rejection, by now depending on claim 10, in which antecedent basis for this element is provided. Applicant respectfully submits that amended claim 11 is now sufficiently definite to meet the requirements of §112, ¶2.

Claims 16 and 17 were each also objected to as unclear regarding which "determining" step was recited at specific locations of these claims; the undersigned notes that these claims were not rejected under §112. To advance the prosecution of this application, claims 16 and 17 are each amended to cancel the objected-to phrases, specifically by canceling the words "the determining step". The responsive actions now recited in each of these claims are simply responsive to the result of the determining. Applicant submits that the amendment presented to claims 16 and 17 is in no way narrowing nor is presented for any reason related to

patentability,<sup>1</sup> presents no new matter, and overcomes the rejection by canceling the objected-to phrases. Applicant further respectfully submits that the informalities noted by the Examiner are no longer present in the claims, and that amended claims 16 and 17 remain sufficiently definite as to meet the requirements of §112, ¶2.

Claim 1 was rejected under §102(b) as anticipated by the Kiuchi et al. reference<sup>2</sup>. To advance the prosecution of this case, and without acquiescing in the §102 rejection, claim 1 is canceled, obviating the rejection.

Claims 2 through 4, and 6 were rejected under §103 as unpatentable over the Kiuchi et al. reference in view of the Osovets reference<sup>3</sup>. Relative to claim 2, the Examiner combined the teachings of the Osovets reference relative to using current and previous addresses of instructions to detect a backward branch with the teachings of the Kiuchi et al. reference, asserting that this combination would be obvious as motivated by the elimination of the repeat instruction taught by Osovets.<sup>4</sup> The claim was rejected accordingly.

Claims 5, 7, and 8 were rejected under §103 as unpatentable over the Kiuchi et al. and Osovets references, and further in view of the George reference<sup>5</sup>. The Examiner asserted that the George reference discloses a valid bit register having bit locations that are each associated with a storage location, and indicating whether the contents of its associated storage location contains a valid instruction code, and that these teachings would be obviously combined with those of the Kiuchi et al. and Osovets references to show whether the entries are valid for accessing to reach claims 5 and 8; the George reference was also asserted as teaching the index register of claim 7.

---

<sup>1</sup> See *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co., Ltd.*, 535 U.S. 722, 62 USPQ2d 1705 (2002), *on remand*, 304 F.3d 1289, 64 USPQ2d 1698 (Fed. Cir. 2002).

<sup>2</sup> U.S. Patent No. 5,579,493, issued November 26, 1996 to Kiuchi et al..

<sup>3</sup> U.S. Patent No. 6,125,440, filed May 21, 1998, and issued September 26, 2000 to Osovets.

<sup>4</sup> Office Action of November 7, 2003, pages 7 and 8, ¶17.

<sup>5</sup> U.S. Patent No. 4,626,988, issued December 2, 1986 to George.

Claim 2 is amended to incorporate the limitations of original claim 1, now canceled. Claim 9 is amended to depend upon amended claim 2, rather than canceled claim 1. No new matter is presented by this amendment to the claims.

Applicant respectfully traverses the §103 rejection of claim 2 and its dependent claims, on the grounds that the combined teachings of the references fall short of the requirements of the claim.

Claim 2 requires, *inter alia*, that the loop cache control logic of the loop cache have a control output coupled to the branch cache register file for controlling writes and reads of the branch cache register file. The claim further requires that the loop cache control logic has an input for receiving a backward branch signal indicating that a fetch address corresponds to a backward branch, and that the loop cache control logic controls the branch cache register file to store an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file. As such, the loop cache control logic enables the storing of instructions in the branch cache register file in response to backward branches, for example as a "loop front cache"<sup>6</sup> or as a "loop tail cache"<sup>7</sup>, and provides the important advantages of minimizing the number of accesses to memory, in a manner that automatically occurs, even for nested program loops, and which can minimize the storing of once-executed instructions in the branch cache register file.<sup>8</sup>

Applicant respectfully submits that the combined teachings of the references fall short of the requirements of claim 2 and its dependent claims, because none of the references disclose loop cache control logic with the functionality recited in claim 2, specifically regarding the control of the branch cache register file to store instruction codes in the event of backward branches and non-matching fetch address.

---

<sup>6</sup> Specification of S.N. 09/713,731, page 14, line 20 *et seq.*; page 20, line 20 through page 21, line 28.

<sup>7</sup> Specification, *supra*, page 22, line 15 *et seq.*; page 32, line 3 through page 33, line 2.

<sup>8</sup> Specification, *supra*, page 5, lines 2 through 12; page 14, lines 20 through 26; page 21, lines 22 through 28; page 32, line 13 through page 33, line 8.

The Kiuchi et al. reference fails to disclose the storing of instruction codes in a branch cache register file in response to a backward branch in combination with a fetch address not corresponding to one of the instruction codes stored in the branch cache register file. Rather, the Kiuchi et al. reference instead discloses storing *every* instruction following a specific repeat instruction in the instruction buffer (at least up to the number of instructions specified in the instruction, or the capacity of the instruction buffer).<sup>9</sup> The Kiuchi et al. reference therefore teaches no storing of instruction codes in response to a backward branch in combination with the non-matching fetch address, as required by claim 2. The Osovets reference also fails to disclose this element of claim 2. Rather, each fetched instruction is stored both in its instruction register and in its shift register bank.<sup>10</sup> The George reference also fails to disclose this recitation of the claim, instead teaching a look-aside buffer or an instruction array into which fetched instructions are simultaneously stored with their storage in an instruction buffer.<sup>11</sup>

Accordingly, the combined teachings of the applied Kiuchi et al., Osovets, and George references fall short of the requirements of claim 2, as none of these references disclose the loop cache control logic as recited in the claim.

Applicant further respectfully submits that there is no suggestion to modify these teachings in such a manner as to reach claim 2 and its dependent claims. The Kiuchi et al. reference, as mentioned in the Office Action,<sup>12</sup> requires the programmer to use a specific "repeat" instruction for its operation; while the Osovets and George references do not require such an instruction, these references simply unconditionally load each fetched instruction into both the instruction buffer and also the shift register bank (Osovets) or instruction array (George), and use the subsequent backward branch to then *access* the instructions in the shift register bank or instruction array. But there is no suggestion from these references to store the instructions in the branch cache register file responsive to the backward branch, as required by claim 2.

---

<sup>9</sup> Kiuchi et al., *supra*, column 6, lines 20 through 45; column 8, lines 21 through 29.

<sup>10</sup> Osovets, *supra*, column 6, line 51 through column 7, line 12.

<sup>11</sup> George, *supra*, column 1, lines 50 through 63; column 4, lines 14 through 21.

<sup>12</sup> Office Action, *supra*, pages 7 and 8, ¶17.c.iv.

The particular advantages provided by the processor of claim 2 include, as mentioned above, the ability to closely cache the instructions for nested program loops, and in minimizing the caching of instructions that are executed only once. This ability is facilitated by the loading of the loop cache after the backward branch has been taken, as performed by the processor of claim 2. As described in the specification, the handling of nested loops by the claimed processor includes both the "loop front cache" example<sup>13</sup> and the "loop tail cache" example<sup>14</sup>. The Kiuchi et al. reference instead teaches the handling of nested loops simply by stacking the loop addresses and other parameters in a "stacking area", which is necessitated by its brute force storing of instruction codes in response to a repeat instruction, and thus does not suggest the solution of claim 2. The other references add nothing to these teachings, because of their unconditional storing of instruction codes as the program is executed. Accordingly, the particular advantages provided by the invention of claim 2 arise directly from the difference between the claim and the prior art, negating suggestion from the prior art to modify their teachings in such a manner as to reach the claim, and indicating the value of the inventive processor.

For these reasons, Applicant respectfully submits that claim 2 and its dependent claims are patentably distinct over the applied references, taken individually or in any proper combination. Applicant therefore respectfully traverses the §103 rejection of claim 2 and its dependent claims.

Applicant also respectfully submits that claim 6 and its dependent claims are further patentably distinct over the applied references. Claim 6 further requires, relative to claim 2 upon which it depends, a next candidate register for storing a fetch address responsive to the loop cache control logic receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file, where the base address register is coupled to the next candidate register, and stores the contents of that next candidate register as the base fetch address responsive to the loop cache control logic receiving a backward branch signal in combination with the fetch address

---

<sup>13</sup> Specification, *supra*, page 21, lines 4 through 9.

corresponding to the next candidate register. In the processor of claim 6 and its dependent claims, the loop cache control logic is recited as controlling the branch cache register file to store an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address corresponding to the contents of the next candidate register. According to this claimed construction, the inventive processor attains the additional advantages of the “loop tail cache” architecture for instruction codes, including the reduced “thrashing” of the instruction cache and also by avoiding the loop caching of instructions that are only executed once.<sup>15</sup>

As discussed above, there is no teaching in the applied prior art of the storing of instruction codes in the branch cache register file in response to a backward branch in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file. Accordingly, there is necessarily no teaching in that prior art of the additional limitations of claim 6 and its dependent claims, especially the next candidate register and the additionally recited functions of the base address register and the loop cache control logic. Furthermore, considering the lack of suggestion of the storing of instruction codes responsive to a backward branch, as discussed above, there is especially no suggestion from the prior art to modify its combined teachings to provide the “loop tail cache” functionality that results from the construction of claim 6 and its dependent claims.

The Examiner asserted, relative to claim 6, that the Kiuchi et al. reference teaches this next candidate register as its element 214b, as responsive to a backward branch signal on its line 113, and that Kiuchi et al. register 214a corresponds to a base address register that stores the contents of the next candidate register as a base fetch address responsive to the loop cache control logic receiving the backward branch signal on line 113 in combination with the fetch address matching the contents of the alleged next candidate register 214b.<sup>16</sup> Applicant respectfully submits that this application of the reference is misplaced. First, the alleged signal 113 of the Kiuchi et al. reference is not a “backward branch” signal, but is instead a signal

---

<sup>14</sup> Specification, *supra*, page 22, line 15 through page 23, line 5; page 32, line 21 through page 33, line 2.

<sup>15</sup> Specification, *supra*, page 22, line 15 through page 23, line 1; page 32, line 21 through page 33, line 2.

<sup>16</sup> Office Action, *supra*, pages 11 and 12, ¶20.

generated responsive to the specific “repeat” instruction inserted into the program code executed by the Kiuchi et al. reference;<sup>17</sup> as discussed above, this repeat instruction begins the unconditional storing of instructions within the instruction buffer 108, and does not constitute a backward branch.<sup>18</sup> In addition, register 214b does not correspond to the next candidate register of the claim, because it does not store a fetch address responsive to the loop cache control logic receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file and because its contents are not used by loop cache control logic and base address register in the manner recited in the claim. Rather, the Kiuchi et al. reference teaches that its register 214b is merely for storing the address of the start of the repeat block, for forwarding to the program counter as a memory address.<sup>19</sup> There is simply no disclosure or suggestion from the Kiuchi et al. reference, nor from the other applied references, of the storing of an instruction code in the branch cache register file responsive to a backward branch signal in combination with the fetch address corresponding to the contents of the next candidate register, in effect upon the second backward branch to the same instruction, as results from the construction of claim 6 and its dependent claims.

For this additional reason, therefore, Applicant submits that claim 6 and its dependent claims are further patentably distinct over the applied references, and accordingly respectfully traverses the §103 rejection of these claims.

Claim 12 and its dependent claims were rejected under §103 as unpatentable over the same references, on similar grounds as discussed above.<sup>20</sup>

Applicant respectfully traverses the rejection of claims 12 through 17, on the grounds that the combined teachings of the applied references fall short of the requirements of claim 12,

---

<sup>17</sup> See Kiuchi et al., *supra*, column 6, lines 8 through 14.

<sup>18</sup> See also Osovets, *supra*, column 2, lines 24 through 44.

<sup>19</sup> Kiuchi et al., *supra*, column 7, lines 17 through 21; column 9, lines 26 through 32; column 12, line 61 through column 13, line 5.

<sup>20</sup> Office Action, *supra*, pages 12 through 17, ¶¶21 through 24; pages 21 through 24, ¶¶ 29 and 30.

and that there is no suggestion to modify these teachings in such a manner as to reach the requirements of the claims.

Claim 12 is directed to a method of fetching instructions for execution by a processor. The recited method includes the step of loading a base location of a branch cache register file with an instruction code corresponding to a fetch address, responsive to a fetch address corresponding to a backward branch operation and not corresponding to valid contents of a loop cache, and setting a valid bit corresponding to that base location. The method further requires the step of then fetching storing instruction codes in a next indexed location of the branch cache register file, and setting a valid bit corresponding to that next indexed location, responsive to receiving fetch addresses following the backward branch operation and within a storage capacity of the branch cache register file. The claimed method then further requires, responsive to receiving fetch addresses corresponding to locations of the branch cache register file for which corresponding valid bits are set, disabling the program memory from performing the read and forwarding the contents of the corresponding branch cache register file.

The method of claim 12 provides important advantages in the operation of programmable processors. According to this method, because instructions are stored in the branch cache register file in response to backward branches that are misses relative to the loop cache, accesses to memory are limited in an automatic manner (*i.e.*, without requiring the programmer to include a “repeat” or other instruction), even for nested program loops.<sup>21</sup> This operation of claim 12 permits implementation of various cache arrangements, including the described examples of a “loop front cache”<sup>22</sup> and a “loop tail cache”<sup>23</sup>.

As discussed above relative to claim 2, the combined teachings of the references fall short of the requirements of claim 12, because none of the references disclose the storing of instruction codes in response to a backward branch in combination with a fetch address not corresponding to one of the instruction codes stored in the branch cache register file. As

---

<sup>21</sup> Specification, *supra*, page 5, lines 2 through 12; page 14, lines 20 through 26; page 21, lines 22 through 28; page 32, line 13 through page 33, line 8.

<sup>22</sup> Specification of S.N. 09/713,731, page 14, line 20 *et seq.*; page 20, line 20 through page 21, line 28.

<sup>23</sup> Specification, *supra*, page 22, line 15 *et seq.*; page 32, line 3 through page 33, line 2.

discussed above, the Kiuchi et al. reference discloses storing *every* instruction following a specific repeat instruction in the instruction buffer (at least up to the number of instructions specified in the instruction, or the capacity of the instruction buffer).<sup>24</sup> The Osovets reference also fails to disclose this step, because it instead discloses storing each fetched instruction is stored both in its instruction register and in its shift register bank.<sup>25</sup> The George reference also fails to disclose this recitation of the claim, instead teaching a look-aside buffer or an instruction array into which fetched instructions are simultaneously stored with their storage in an instruction buffer.<sup>26</sup> Accordingly, Applicant respectfully submits that none of the applied references discloses or suggests the loading and storing steps recited in claim 12.

The Examiner also asserted that the Osovets reference teaches the use of valid bits in the manner recited by claim 12.<sup>27</sup> Specifically, the Examiner asserted that because the addresses stored are for valid instructions, there must be at least one bit for keeping track of the validity information.<sup>28</sup> Applicant respectfully traverses the §103 rejection, to the extent based on this assertion, on the grounds that there are no such teachings in the Osovets reference.

The “valid instructions” mentioned in the Osovets reference refer simply to those instructions stored in its shift register bank.<sup>29</sup> The determination of whether an instruction is “valid” according to the Osovets reference depends on whether its address is within or outside of the range of valid instructions stored in that shift register bank.<sup>30</sup> There is simply no disclosure of a “valid” bit for the shift register file of the Osovets reference; this is consistent with the construction and operation of the Osovets file as a shift register, in which contents are shifted along as the additional instructions are loaded into the shift register bank. On the other hand, the method of claim 12 not only determines whether the fetch address corresponds to a location of the branch cache register file, but also whether the contents of that location are valid, prior to effecting the disabling and forwarding steps recited in the claim.

---

<sup>24</sup> Kiuchi et al., *supra*, column 6, lines 20 through 45; column 8, lines 21 through 29.

<sup>25</sup> Osovets, *supra*, column 6, line 51 through column 7, line 12.

<sup>26</sup> George, *supra*, column 1, lines 50 through 63; column 4, lines 14 through 21.

<sup>27</sup> Office Action, *supra*, page 13, ¶21.b.iii.

<sup>28</sup> *Id.*

<sup>29</sup> Osovets, *supra*, column 9, lines 44 through 46, and lines 60 through 64;

Applicant therefore submits that the combined teachings of the applied Kiuchi et al., Osovets, and George references fall short of the requirements of claim 12, as none of these references disclose the loading, fetching, and storing steps that are responsive to a backward branch operation, as recited in the claim. For this reason, and also because the Osovets reference fails to disclose responsive operations relative to the valid bit as claimed, contrary to the assertion by the Examiner, Applicant traverses the rejection of claim 12 and its dependent claims.

Applicant further respectfully submits that claim 12 and its dependent claims are patentably distinct over the applied references, on the grounds that there is no suggestion from the prior art to modify their teachings in such a manner as to reach the claim. As asserted above relative to claim 2, the Kiuchi et al. reference relies upon a specific “repeat” instruction for its loading of an instruction buffer; while the Osovets and George references do not require such an instruction, these references simply unconditionally load each fetched instruction into both the instruction buffer and also the shift register bank (Osovets) or instruction array (George), and use the subsequent backward branch to then *access* the instructions in the shift register bank or instruction array. These references simply lack any suggestion to load instructions into branch cache register file responsive to a backward branch and to the fetch address not corresponding to valid cache contents, as required by claim 12.

Further, Applicant submits that the particular advantages provided by the method of claim 12 support the patentability of these claims. As discussed above, the claimed method enables the efficient caching of instructions, even for nested program loops, and also enables the minimizing the caching of instructions that are executed only once. These advantages are accomplished by the loading of the branch cache register file after a backward branch has been taken, for example as performed by both the “loop front cache” example and the “loop tail cache” example described in the specification.<sup>31</sup> In contrast, the Kiuchi et al. reference teaches the handling of nested loops simply by stacking the loop addresses and other parameters in a

---

<sup>30</sup> Osovets, *supra*, column 9, line 64 through column 10, line 5.

<sup>31</sup> Specification, *supra*, page 21, lines 4 through 9; page 22, line 15 through page 23, line 5; page 32, line 21 through page 33, line 2.

"stacking area", and the other references also teach only the unconditional storing of instruction codes as the program is executed. Accordingly, the particular advantages provided by the method of claims 12 through 17 directly result from this difference, supporting the patentability of these claims.

Applicant therefore respectfully submits that claim 12 and its dependent claims are patentably distinct over the applied references, taken individually or in any proper combination.

Claim 15 further requires, relative to claim 12 upon which it depends, detecting whether the received fetch address corresponds to a backward branch relative to a previous fetch address; if so, the method requires comparing the received fetch address with the contents of a candidate register, and if not, the method requires storing the received fetch address in the candidate register. The step of loading the base location of a branch cache register file is then performed responsive to the comparing step detecting that the received fetch address matches the contents of the candidate register. In this way, the method of claim 15 provides the "loop tail cache" operation described in the specification, and provides the important advantages of such an operation, including the limiting of "thrashing" of the instruction cache.<sup>32</sup>

Applicant respectfully submits that claim 15 is further patentably distinct over the applied references. The shortfall of the combined teachings of the Kiuchi et al., Osovets, and George references from claim 12 necessarily result in these teachings falling further short of the requirements of claim 15, in which the loading step is performed responsive to the fetch address corresponding to the contents of the candidate register, in which the candidate register stores a previous fetch address. There is even less suggestion from the prior art to modify these teachings to reach claim 15, considering that the references fail whatsoever to disclose the loading of a branch cache register file in response to backward branches. Furthermore, as discussed above relative to claim 6, there is simply no "candidate" register disclosed in the Kiuchi et al. reference; the register alleged by the Examiner in this regard is simply a register used to build a program counter value for fetching the top address again.

Applicant therefore respectfully submits that claim 15 and its dependent claims 16 and 17 are further patentably distinct over the applied references. The §103 rejection of these claims 15 through 17 is therefore traversed, for this additional reason.

The prior art cited by the Examiner but not applied has been considered, but is not felt to come within the scope of the claims in this case.

For the above reasons, Applicant respectfully submits that all claims now in this case are in condition for allowance. Reconsideration of the above-referenced application is therefore respectfully requested.

Respectfully submitted,



Rodney M. Anderson

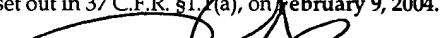
Registry No. 31,939

Attorney for Applicants

Anderson, Levine & Lintel, L.L.P.  
14785 Preston Road, Suite 650  
Dallas, Texas 75254  
(972) 664-9554

CERTIFICATE OF MAILING  
37 C.F.R. 1.8

The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, and addressed as set out in 37 C.F.R. §1.1(a), on February 9, 2004.



Rodney M. Anderson  
Registry No. 31,939

---

<sup>32</sup> Specification, *supra*, page 22, line 1 through page 23, line 5; page 32, line 21 through page 33, line 2.

JCIIS  
12/2004  
10

TI-29298  
1 of 5

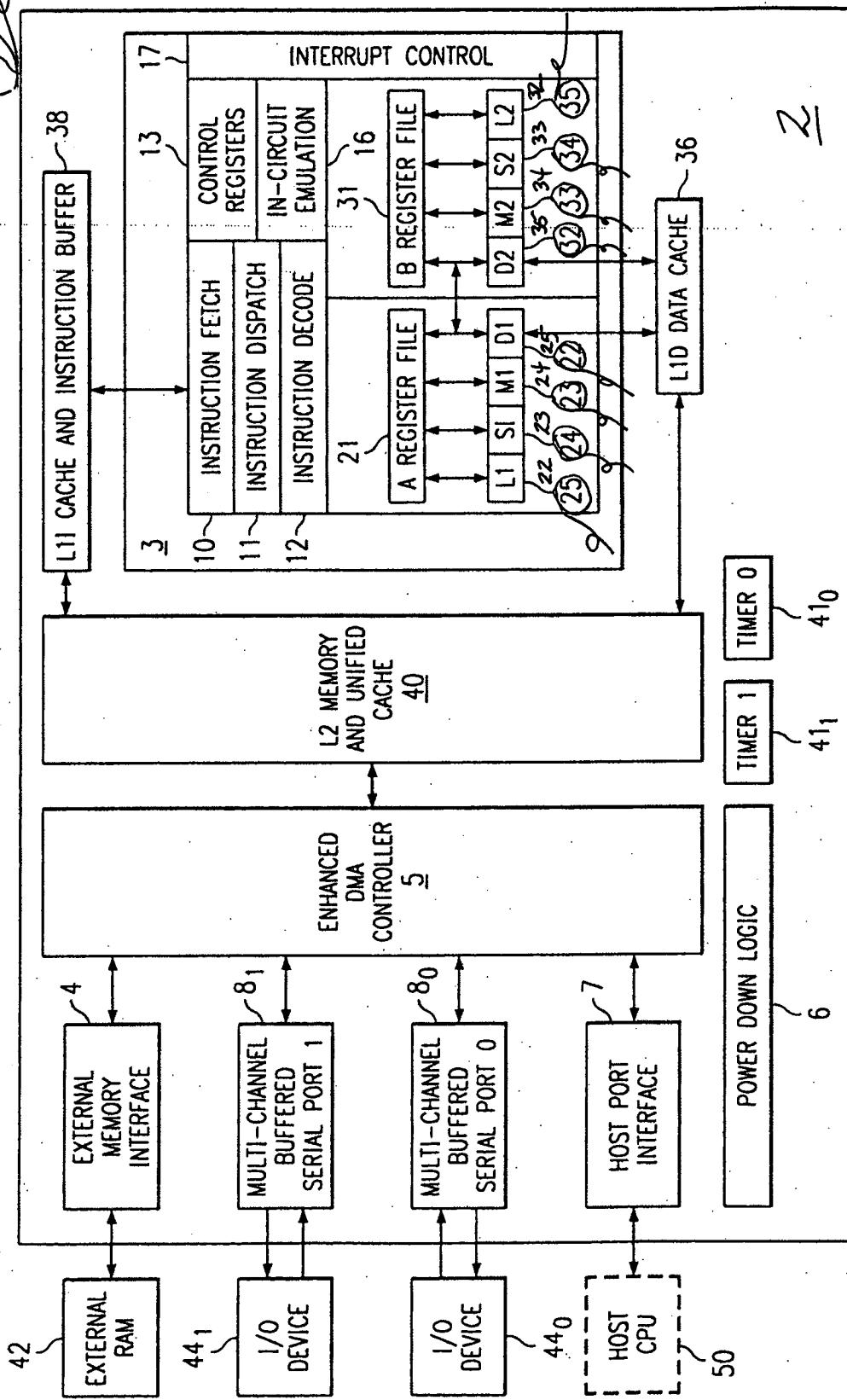


FIG. 1